

RAPPORT DE STAGE
Savin Rémy

Développement Web et Web Mobile

Maitre de Stage :
Mr Buelens Geoffray

Responsable Pédagogique :
Mr Anthony Cosson

Ecole Nationale d'Informatique
Chartres-De-Bretagne

Promotion : *D2WM05*
Stage du 24/06/19 au 26/08/19

Table des matières

| | |
|--|----|
| 1 Remerciements | 3 |
| 2 Liste des compétences | 4 |
| 3 Projet et Objectif | 5 |
| 4 Environnement de travail | 6 |
| 5 Résumé du projet | 7 |
| 6 Cahier des charges | 8 |
| 6.1 Présentation d'ensemble du projet | 8 |
| 6.1.1 Présentation de l'entreprise | 8 |
| 6.1.2 Les objectifs du site | 8 |
| 6.1.3 La cible adressée par le site | 8 |
| 6.1.4 Périmètre du projet | 8 |
| 6.2 Fonctionnalité du site: | 9 |
| 6.2.1 Coté Utilisateur | 9 |
| 6.2.2 Coté Admin | 9 |
| 6.2.3 Diagramme UseCase | 10 |
| 6.3 Description graphique et ergonomique | 11 |
| 6.3.1 Charte graphique | 11 |
| 6.3.2 Inspirations | 11 |
| 6.3.3 Photo | 11 |
| 6.4 Maquettage | 12 |
| 6.4.1 Vue Desktop | 12 |
| 6.4.2 Vue Mobile | 13 |
| 6.5 Modèle | 14 |
| 6.5.1 Modèle MCD | 14 |
| 6.5.2 Modèle MLD | 15 |
| 7 Spécifications techniques et sécurité | 16 |
| 7.1 Langage | 16 |
| 7.1.1 Langage principal et Framework associé | 16 |
| 7.1.2 Autres langages utilisés | 17 |
| 7.2 Base de données | 17 |
| 7.2.1 MySQL | 17 |
| 7.3 Terminal | 17 |
| 7.3.1 Cmder | 17 |
| 7.4 Autre outil | 18 |
| 7.4.1 Git Kraken | 18 |
| 7.5 Gestionnaire de dépendance | 18 |
| 7.5.1 Composer | 18 |
| 7.6 Bundle | 18 |
| 7.6.1 Les bundles installés | 18 |
| 8 Présentation de la fonctionnalité la plus significative | 21 |
| 8.1 La commande | 21 |
| 8.1.1 Le panier | 21 |
| 8.1.2 Sécurité | 22 |
| 8.1.3 Le Controller | 22 |
| 8.1.4 Le futur de cette fonctionnalité | 23 |
| 9 Réalisation | 24 |

| | |
|---|-----------|
| 9.1 Front-end | 24 |
| 9.1.1 La NavBar | 24 |
| 9.1.2 Le Footer | 24 |
| 9.1.3 JQuery | 24 |
| 9.2 Front-End & Back-End..... | 25 |
| 9.2.1 Connexion | 26 |
| 9.2.2 Mot de passe oublié | 27 |
| 9.2.3 Profil..... | 28 |
| 9.2.4 Modifier profil | 29 |
| 9.2.5 Contact..... | 30 |
| 9.3 Autre..... | 31 |
| 9.3.1 Photo..... | 31 |
| 10 Présentation du jeu d'essai | 33 |
| 10.1 Inscription | 33 |
| 10.1.1 Création du Controller | 34 |
| 10.1.2 Création du formulaire | 34 |
| 10.1.3 Préparation du Controller | 35 |
| 10.1.4 Le ReCAPTCHA | 36 |
| 10.1.5 Encodage et envoi en base de données | 36 |
| 11 Sécurité | 38 |
| 11.1 Le reCAPTCHA | 38 |
| 11.1.1 Pourquoi installer ReCAPTCHA..... | 38 |
| 11.1.2 Choix du type de reCAPTCHA | 38 |
| 11.1.3 Installation..... | 38 |
| 11.2 Le hashage | 39 |
| 11.2.1 Pourquoi hasher le mot de passe..... | 39 |
| 11.2.2 Comment sécuriser le mot de passe | 39 |
| 11.3 Les expressions régulières..... | 39 |
| 11.3.1 Définition..... | 39 |
| 11.3.2 Choix d'utilisation | 40 |
| 12 Veille effectuée..... | 41 |
| 12.1 Recherche | 41 |
| 12.1.1 Contexte | 41 |
| 12.1.2 Phase de recherche | 41 |
| 12.2 Traduction | 41 |
| 12.2.1 Version originale | 41 |
| 12.2.2 Version traduite..... | 42 |
| 13 Conclusion | 44 |
| 14 Annexes | 45 |

1 Remerciements

A Mr BULENS Geoffrey, patron de l'entreprise qui nous a permis de trouver un stage. Il a toujours été disponible en cas de besoin et était à l'écoute sur nos recommandations.

A Geoffrey Oliver, avec qui j'ai travaillé sur ce projet passionnant pendant deux mois, pour son professionnalisme et son soutien. Nous avons traversé ensemble les difficultés et nous ne laissons aucun de nous bloquer sur une problématique. Ces deux mois ont été intense, mais je n'aurai pu espérer mieux comme collègue de travail.

A mes collègues de promotion, professeurs et anonymes du net qui m'ont permis de débloquer certaines situations et qui m'ont soutenu pendant tout ce stage.

A Léa, qui m'a supporté pendant cette période et qui m'a apporté son soutien pendant toute la formation et le stage.

A Tristan Dréau, qui m'a apporté son aide durant ce stage et qui m'a prodigué de précieux conseils.

2 Liste des compétences

Nom et prénom du candidat : SAVIN Rémy

Document complété d'un commun accord entre le stagiaire et le responsable du stage en entreprise à joindre au rapport d'activité.

| Compétences Voir le détail dans le référentiel d'emploi, d'activités et de compétences | Cocher les compétences mises en œuvre lors du projet en entreprise (en totalité ou partiellement) |
|--|--|
| C1 - Maquetter une application | X |
| C2 - Réaliser une interface utilisateur web statique et adaptable | X |
| C3 - Développer une interface utilisateur web dynamique | X |
| C4 - Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce | X |
| C5 - Créer une base de données | X |
| C6 - Développer les composants d'accès aux données | X |
| C7 - Développer la partie back-end d'une application web ou web mobile | X |
| C8 - Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce | X |


Observations éventuelles :

Du stagiaire :

Projet très intéressant pour toutes les fonctionnalités à mettre en place. Ce projet est long à réaliser car nous devons être vigilant sur les éventuelles failles de sécurité. Nous voulions également faire une application web propre et moderne ce qui nous a pris beaucoup de temps pour obtenir quelque chose de correct. Désormais, les futurs développeurs qui reprendront ce projet n'auront aucun mal à reprendre là où nous nous sommes arrêtés.

De l'entreprise : *TRES bon travail, autonome très studieux*

Nom et Prénom du stagiaire :
SAVIN REMY

Signature : 

Entreprise :
N.L.B.

Nom du responsable de stage :
Buelens

Signature : **N.L.B.** 

4 Rue du Gast
35700 Rennes
Tél. : 06.66.06.49.74
EURL au Capital de 3000 €
Siret 819 175 233 00017 - APE 5610C

3 Projet et Objectif

Mr BULENS nous a fait part de son souhait d'avoir son propre site internet pour son Food-truck.

Actuellement, les réservations pouvaient se faire en envoyant un SMS sur le portable professionnel de Mr BULENS. Cependant, il n'était visible que localement et seuls les habitués réservaient.

La création du site permettrait de fluidifier les personnes commandant en même temps et d'être répertorié sur internet lorsque quelqu'un recherche un Food-Truck sur Rennes.

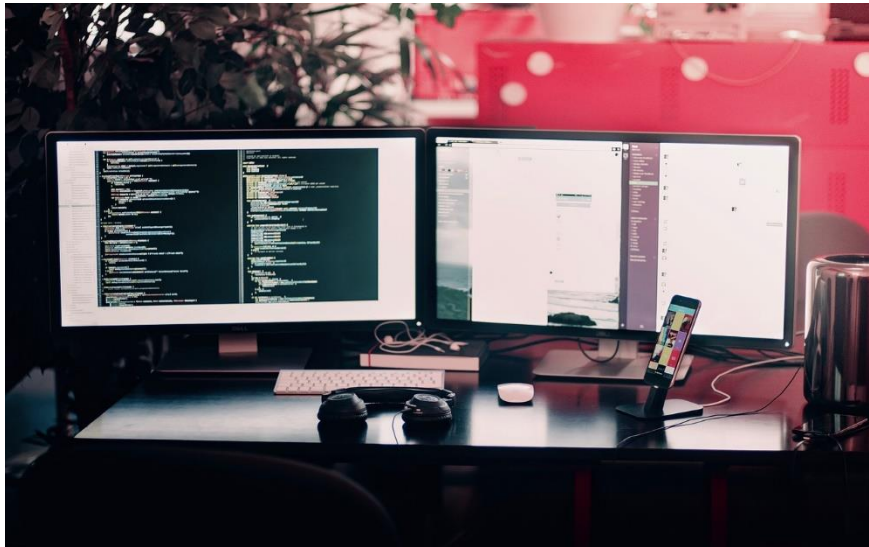
Enfin, le site permettra d'augmenter les bénéfices de l'entreprise grâce à l'ajout de suppléments.

Son souhait est que nous nous inspirions des couleurs de la Bretagne et Flamandes pour le design du site.

Afin d'éviter des surcoûts et éviter des commandes « fantôme », il n'a pas souhaité mettre en place une option de paiement en ligne. Le paiement se fera au camion.

Mr BULENS a pour projet dans l'avenir d'avoir plusieurs camions, voir un restaurant de vente à emporter.

4 Environnement de travail



L'entreprise ne possède pas de support technique, ni de personne ayant des connaissances dans le langage informatique.

N'ayant pas de locaux à nous proposer, nous avons travaillé à domicile pendant ces deux mois. Mon collègue de travail et moi-même avons chacun un ordinateur personnel, capable de réaliser l'application web.

Nous communiquons tous les jours en cas de besoin via Skype ou Discord où lorsque certaines parties du site, design ou fonctionnalité, nécessitaient un deuxième avis. En effet, ces outils permettent de partager son écran à une autre personne.



De plus, n'ayant pas de support technique, nous avons dû effectuer les recherches adéquates afin de résoudre les problèmes que nous rencontrions. Pour cela, nous nous sommes aidés de la documentation officielle de Symfony. En cas de problème plus complexe, nous allions sur des forums spécialisés dans le développement web.

5 Résumé du projet

L'entreprise « Oh Ch'ti Breizh » est un camion de type Food-truck dont la principale activité est la restauration rapide. Mr Buelens, gérant de l'entreprise, recherchait des personnes pouvant créer son site e-commerce pour fluidifier l'arrivée des clients. Pour cela, nous lui avons proposé une application web permettant au client de réserver en avance sa commande.

Dans un premier temps, nous avons fait un recueil de données auprès de lui. Il souhaite mettre en avant la possibilité de faire de l'évènementiel, pour des mariages, CE et autres. Le site lui permettra également d'être d'avantage visible sur internet. Le site web doit rester le plus simple possible. Entre autres, une navigation claire et des articles bien mis en valeur. Il n'a pas souhaité d'option de paiement en ligne dans un premier temps. Le but principal est la réservation et la prise de contact. La deuxième étape a été la conception technique du site internet. Il a donc fallu concevoir différents schémas comme le diagramme de cas d'utilisation ainsi que le modèle conceptuel de données. Après la conception, nous avons mis en place les fonctionnalités primaires du site comme l'inscription, la connexion ou le panier par exemple. Par la suite, il souhaitait pouvoir changer les informations sur le site web comme les tarifs, les menus etc... Nous avons donc mis en place le bundle EasyAdmin (définition p.16) afin qu'il puisse modifier les informations comme il le souhaite.

A l'heure actuelle, il reste plusieurs fonctionnalités à mettre en place comme l'affichage de la commande par exemple. Le projet sera présent sur GitHub avec ce même rapport de stage qui comprendra, en annexe p. 45, un document à l'intention des futurs stagiaires/développeurs pour la poursuite du projet.

6 Cahier des charges

6.1 Présentation d'ensemble du projet

6.1.1 Présentation de l'entreprise

- Oh Ch'ti Breizh a été créée le 18 mars 2016. C'est une entreprise de restauration rapide. Elle propose des plats typiques de la Région Bretonne et Flamande.
L'entreprise est dirigée par Mr Buelens. Les clients peuvent aller au camion pour commander et être servi ou commander par sms et venir récupérer leur repas.
L'entreprise possède un site "wix" gratuit mais le design du site et les fonctionnalités possibles sont très limitées. De plus, les pubs pour l'entreprise "wix" sont très présentes sur le site.
- A terme, l'entreprise souhaiterait ouvrir un restaurant de vente à emporter.

6.1.2 Les objectifs du site

- Fluidifier les réservations et éviter les longues files d'attente.
- L'entreprise pourra également signaler sur le site internet en cas d'empêchement via un message à destination des utilisateurs. Il apparaîtra sur la page d'accueil du site.
- De plus, l'entreprise sera d'avantage visible autrement que par ses emplacements habituels.
- Le site devra permettre également d'attirer d'avantage de client et augmenter les profits de l'entreprise.

6.1.3 La cible adressée par le site

- "Oh Ch'ti Breizh" cible les employés et personnes en formation. Ces personnes ont peu de temps pour se restaurer et le site internet leur permettra d'être servi plus rapidement.
- Le site internet permettra d'attirer une population plus jeune, adepte de la restauration rapide.
- Les personnes souhaitant organiser un évènement pourront faire appel, via le site internet, aux services de l'entreprise.

6.1.4 Périmètre du projet

- L'activité vise uniquement la France métropolitaine
- Le site sera disponible en français uniquement
- Le site sera intégralement responsive

6.2 Fonctionnalité du site:

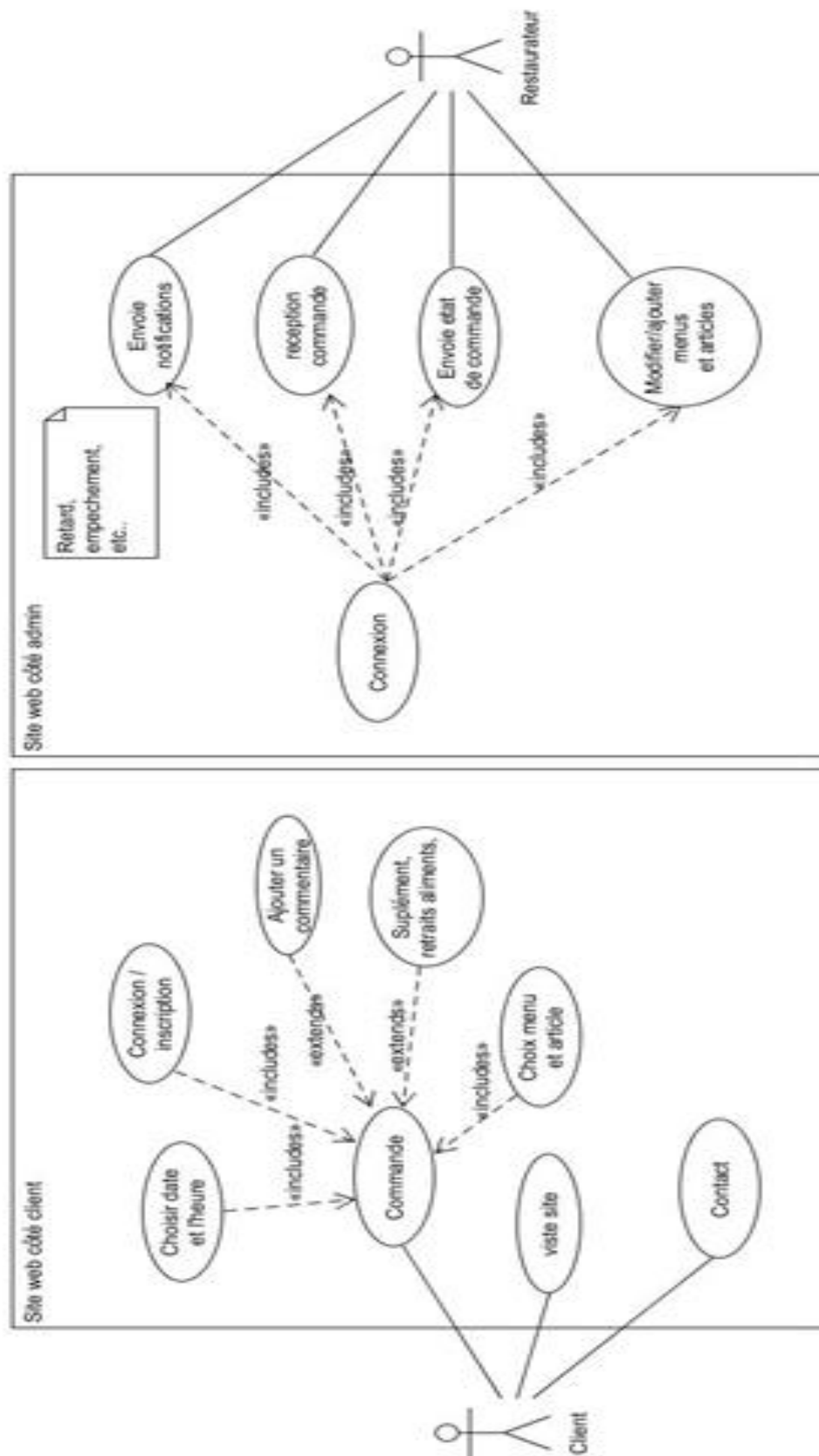
6.2.1 Coté Utilisateur

| L'utilisateur non connecté pourra: | L'utilisateur connecté pourra en plus: |
|--|---|
| <ul style="list-style-type: none">• Envoyer un message | <ul style="list-style-type: none">• Accéder à son profil et le modifier si besoin (sauf pseudo, nécessaire pour se connecter) |
| <ul style="list-style-type: none">• Voir la carte | <ul style="list-style-type: none">• Voir sa commande via un onglet qui apparaîtra lorsqu'il sera connecté |
| <ul style="list-style-type: none">• Ajouter des articles dans son panier | <ul style="list-style-type: none">• Valider son panier |
| <ul style="list-style-type: none">• Ajouter des suppléments, retirer des ingrédients | |
| <ul style="list-style-type: none">• Voir son panier et choisir l'heure de retrait de son choix | |
| <ul style="list-style-type: none">• S'inscrire | |

6.2.2 Coté Administrateur

| L'administrateur pourra après connexion: |
|--|
| <ul style="list-style-type: none">• Réceptionner les commandes |
| <ul style="list-style-type: none">• Modifier l'état des commandes:<ul style="list-style-type: none">- En attente- En cours- Terminée- En attente de réception |
| <ul style="list-style-type: none">• Envoyer des messages à l'intention des utilisateurs |
| <ul style="list-style-type: none">• Modifier/ajouter des menus et articles |

6.2.3 Diagramme UseCase



6.3 Description graphique et ergonomique

6.3.1 Charte graphique

La charte graphique doit respecter au maximum les consignes suivantes :

- La charte graphique doit être moderne et épurée.
- Le site devra utiliser les couleurs présente sur le camion : Noir, Rouge, Bleu, Jaune, Blanc
- Utiliser les inspirations ci-dessous

6.3.2 Inspirations

Inspirations graphiques métiers

- <https://salvanatural.fr/>
- <http://www.lecamionacroquer.fr>

Inspirations graphiques généralistes

- <https://www.pizzahut.fr/>

Les inspirations ont été diverses et variées. Le but étant de voir les fonctionnalités/design des autres food-truck aux alentours. Cependant pour la plupart, ils ne proposaient qui ? pas de vente en ligne mais juste l'affichage de la carte.

6.3.3 Photo

Les photos seront réalisées par l'équipe en charge du développement. Elles devront éventuellement être retouchées via un logiciel afin d'améliorer et promouvoir les produits de "Oh Ch'ti Breizh". Elles seront prises majoritairement au camion.

6.4 Maquettage

6.4.1 Vue Desktop



En vue Desktop, le site doit être agréable à lire et moderne. La page d'accueil présentera l'entreprise, son positionnement tout au long de la semaine ainsi que les cartes du Food-Truck. Ces dernières doivent être de taille suffisamment grande pour les lire sans nécessité de zoomer. Enfin, sur cette page, les informations de l'entreprise seront présentes et l'utilisateur pourra contacter l'entreprise via un bouton cliquable.

6.4.2 Vue Mobile



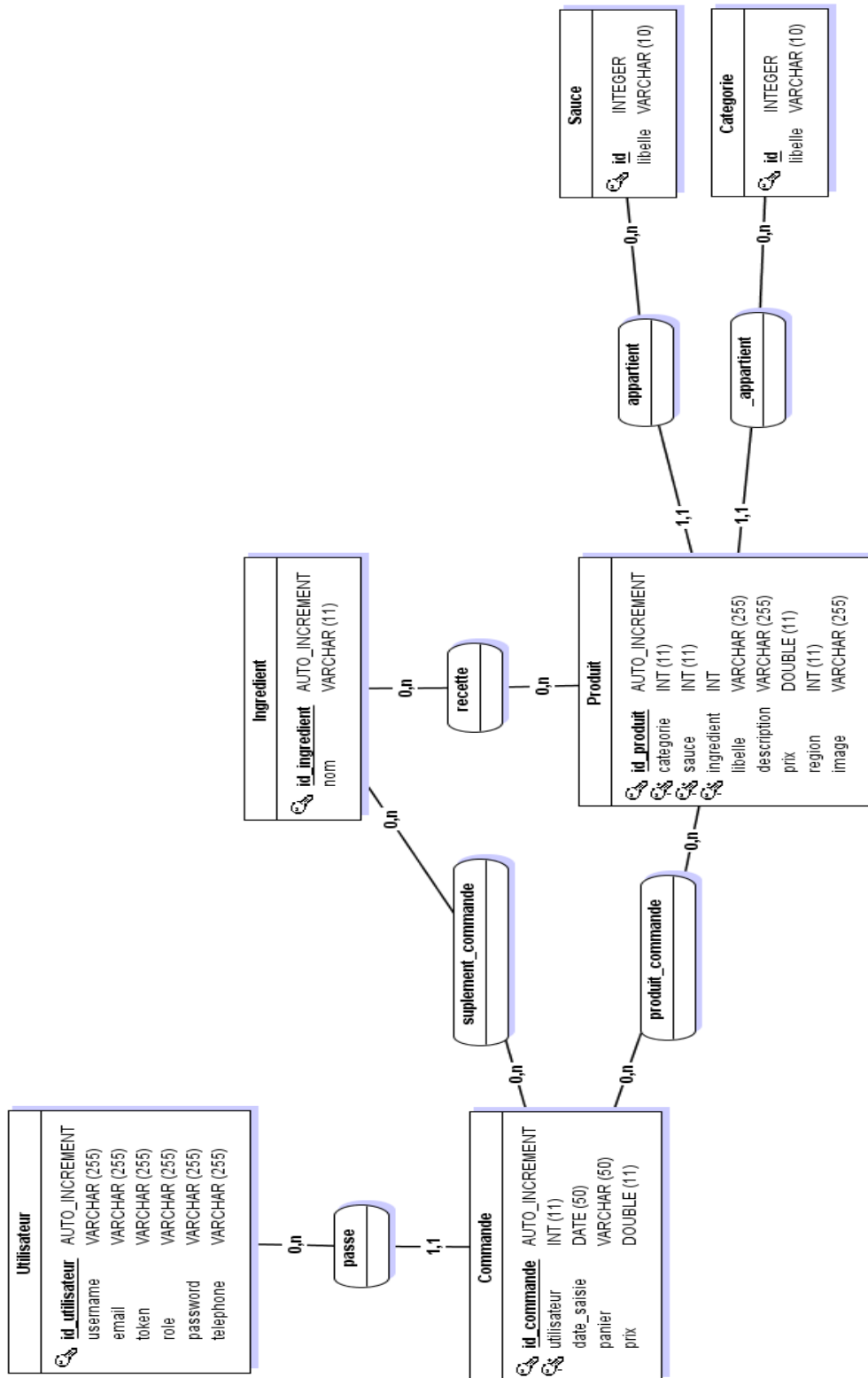
Les images et articles doivent être lisibles même sur petit écran.

Lorsque l'écran est en dessous de 1024px, les boutons de la NavBar disparaissent et deviennent un burger.

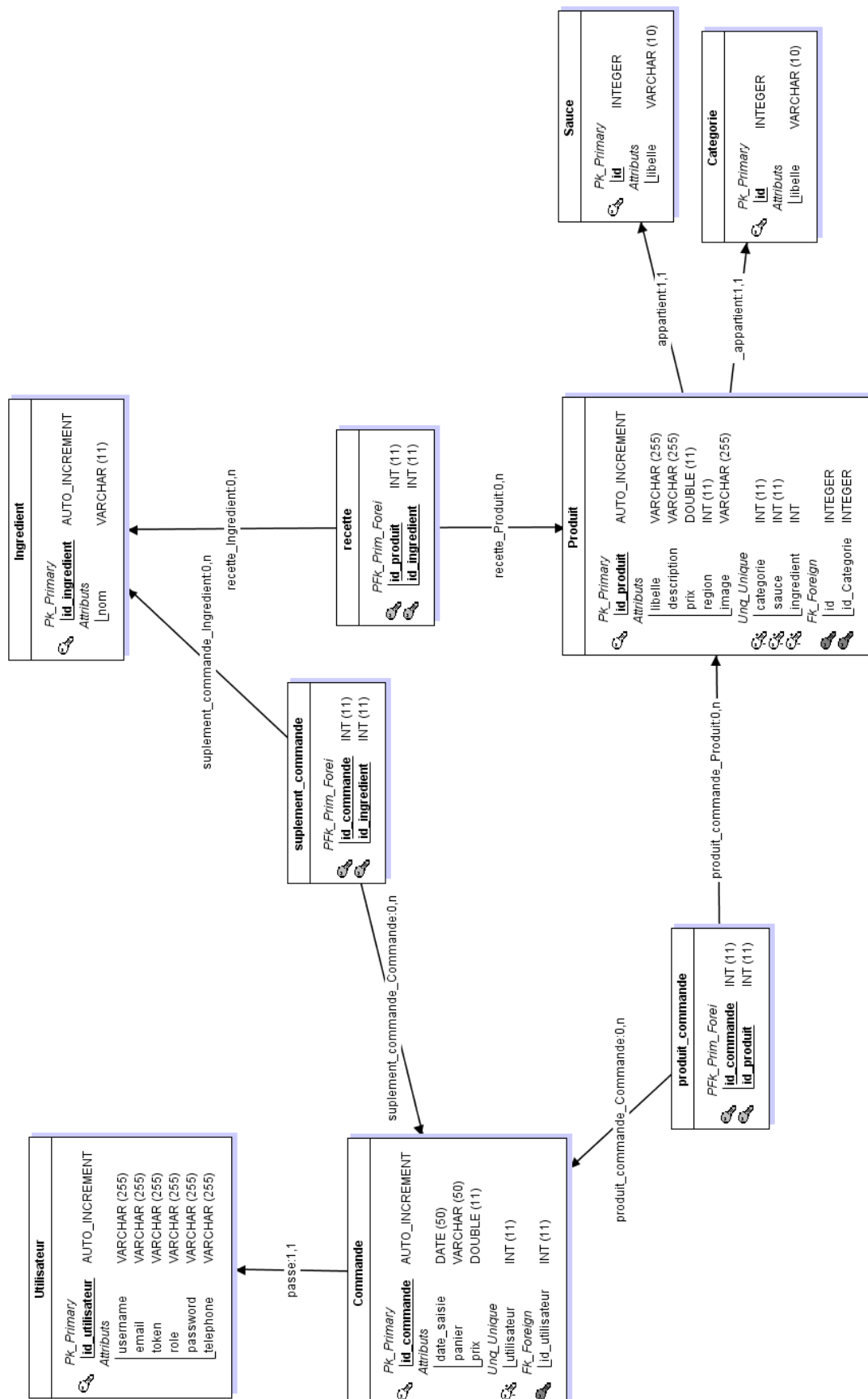
Les articles apparaîtront les uns à la suite des autres.

6.5 Modèle

6.5.1 Modèle MCD



6.5.2 Modèle MLD



7 Spécifications techniques et sécurité

7.1 Langage

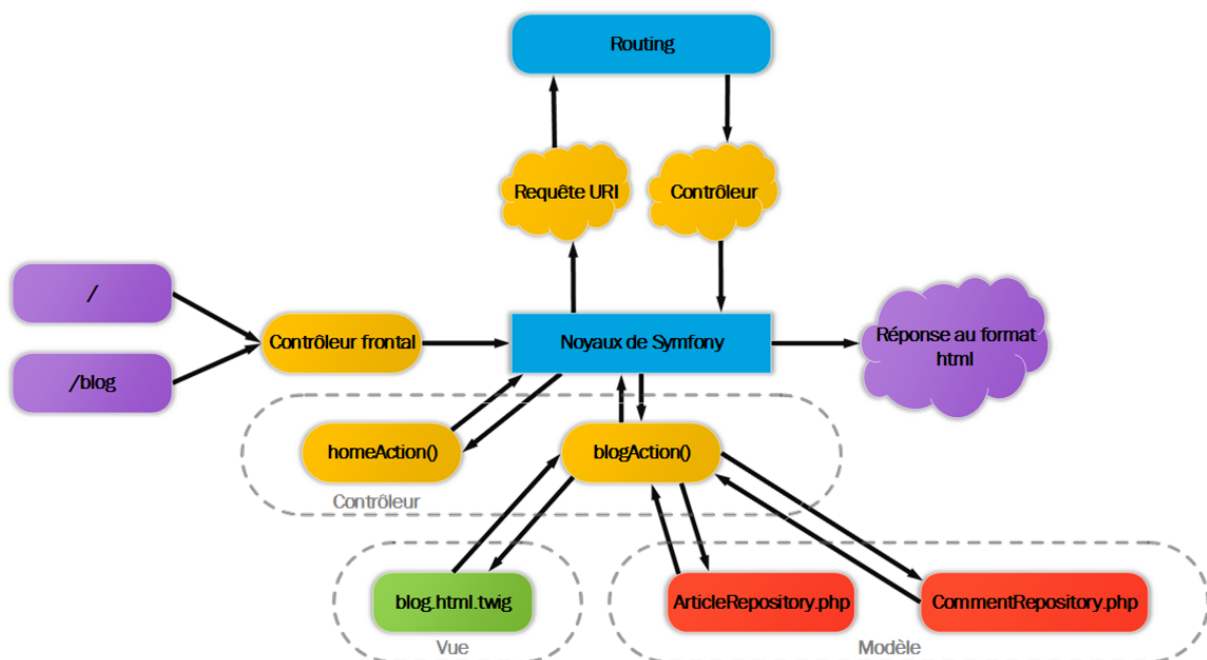
7.1.1 Langage principal et Framework associé

Le choix du langage s'est finalement porté sur PHP. Nous avons vu ce langage lors de la formation. Il est accessible sur la plupart des systèmes d'exploitation et est très populaire. Principalement utilisé pour produire des pages Web dynamique, il correspondait parfaitement au projet. De nombreux sites connus ont été réalisés via le langage PHP, notamment Facebook. Ce qui montre une certaine robustesse de ce langage.

De plus, nous avons utilisé le Framework Symfony 4. Nous l'avons vu lors de la formation à L'ENI et nous avons tout de suite été conquis par la prise en main et les possibilités proposées. Il permet un bon apprentissage, en plus d'être complet et très performant. Cela permet une structure de type MVC, permettant ainsi d'organiser le code et de le rendre ainsi plus évolutif et maintenable. Enfin, Symfony est considéré comme le Framework PHP le plus puissant et le plus flexible. En cas d'ajout de fonctionnalité, de futurs développeurs n'auront aucun mal à comprendre cette structure et pourront, par exemple, ajouter la gestion de paiement. De plus, le Framework Symfony a une très grande communauté, est Open Source et évolue sans cesse grâce à cette dernière. J'ajouterai que Symfony est une création Française créée par M. Potencier Fabien.



Le choix s'est donc porté naturellement sur ce Framework au vu des avantages qu'il offre. Concernant l'IDE, nous avons choisi PHPStorm, un éditeur PHP, HTML, CSS et javascript édité par JetBrains. Enfin, nous avons installé comme Framework: Symfony 4.



Source : <http://igm.univ-mlv.fr/~dr/XPOSE2014/Symfony/structure.html>

7.1.2 Autres langages utilisés

Bootstrap : Bootstrap est une boîte à outils open source pour le développement avec HTML, CSS et JS. Elle a été très utile pour son système de grid permettant une application responsive pour les petits écrans de manière très simple.

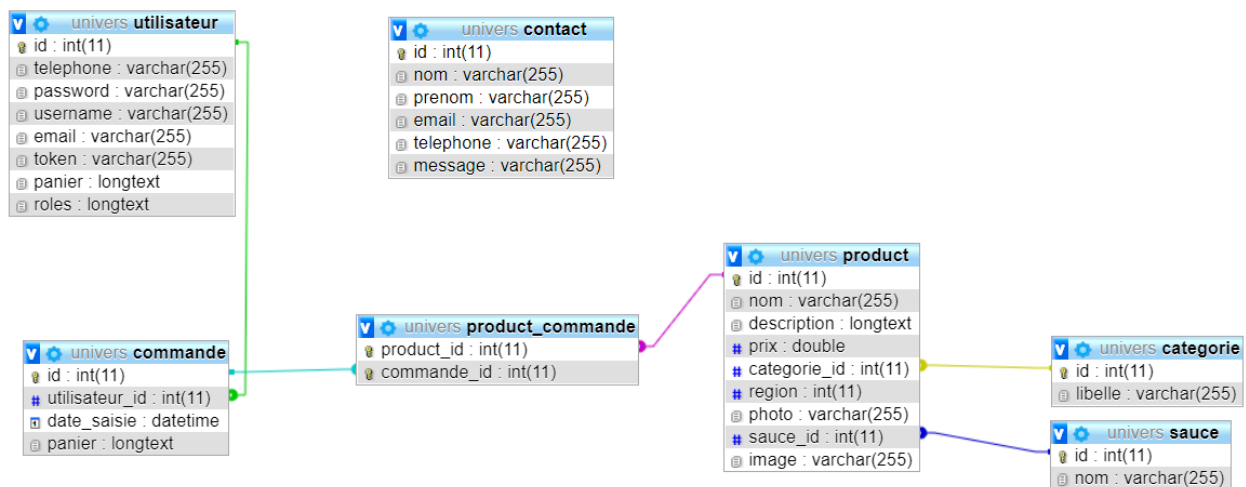
JavaScript : c'est un langage de programmation qui permet d'ajouter de l'interactivité à notre application web. Nous avons utilisé plus particulièrement JQuery, qui est une bibliothèque javascript qui facilite l'écriture de script coté client (exemple : la page « menu » possède 4 boutons et permet d'afficher les articles correspondant sans rechargement de la page grâce aux requêtes asynchrones).

7.2 Base de données

7.2.1 MySQL

Durant la phase de développement, il est nécessaire de choisir quel système de gestion de base de données utiliser.

Notre choix s'est porté sur MySQL. Ce SGBD est gratuit, léger et performant. Il fera le lien avec notre application grâce à Doctrine et notre terminal **Cmdr**.



7.3 Terminal

7.3.1 Cmdr

Notre choix s'est porté sur **Cmdr**. Nous l'avons vu lors de notre formation également. Il est basé sur l'émulateur de console « ConEmu ». Nous pouvions donc l'installer très facilement. De plus il possède des fonctionnalités très intéressantes comme :

- L'intégration des principales commandes linux
- Intégration de git

- Pouvoir observer un mini 'git statut' avec le nombre de fichier à commit/push
- Une interface agréable à l'œil

7.4 Autre outil

7.4.1 Git Kraken

Enfin, pour mettre en relation nos travaux respectifs, nous avons utilisé Git Kraken qui est un logiciel de gestion de versions décentralisé basé sur git. Cela nous a permis de travailler à distance et de mettre en commun nos travaux respectifs.



7.5 Gestionnaire de dépendance

7.5.1 Composer

Nous avons choisi d'utiliser Composer. Il sert de gestionnaire de dépendance entre application et librairie.

Grâce à cet outil, nous pouvons installer des librairies utilisables pour notre projet et choisir leurs versions.

```
C:\wamp64\www\hope>composer req doctrine
Using version ^1.0 for symfony/orm-pack
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Restricting packages listed in "symfony/symfony" to "4.2.*"
Nothing to install or update
Writing lock file
Generating autoload files
ocramius/package-versions: Generating version class...
ocramius/package-versions: ...done generating version class
Executing script cache:clear [OK]
Executing script assets:install public [OK]
```

Toutes les informations seront enregistrées dans le « Composer.json ».

7.6 Bundle

7.6.1 Les bundles installés

EasyAdmin : Ce bundle est un ensemble d'outils permettant de créer une interface d'administration sécurisée facilement. Il possède également une documentation très complète.

Il supporte Symfony 2, 3 et 4.

Il permet de :

- Réaliser des opérations SCRUD (Search/Create/Read/Update/Delete) facilement sur nos entités Doctrine.
- Trier les informations

Enfin, sa rapidité et sa performance permettent une navigation fluide.

VichUploader : Ce bundle permet de gérer l'upload de fichier. Il possède deux composants :

- FileNamer qui permet de gérer le nommage du fichier

- DirectoryNamer qui permet d'agir sur le nom du sous-dossier dans lequel le fichier sera déposé.

SwiftMailer : C'est une librairie, orientée objet, permettant l'envoi de mail facilement et rapidement. Cet envoi de mail est possible après configuration du server smtp. Notre maitre de stage possédant une adresse Gmail, nous avons donc utilisé cette dernière pour la configuration de SwiftMailer.

```
// grâce à SwiftMailer on prépare le mail qui sera envoyé: expéditeur, destinataire etc...
$envoi = (new \Swift_Message('Message'))
    ->setFrom($mail)
    ->setTo("remy_savin@hotmail.fr")
    ->setBody(
        $messageEnvoi,
        'text/html');
// on envoie le mail avec message pour informer l'utilisateur
$mailer->send($envoi);
$this->addFlash('notice', 'Message envoyé');
```

Security : Dans notre projet, nous avons utilisé ce bundle afin de permettre la création du fichier « security.yaml » . Nous avons pu, de ce fait, gérer l'authentification, les chemins autorisés ou non pour chaque Rôle, ainsi que le type d'encodage du password de l'utilisateur.

Twig : Moteur de Template proposé par Symfony. Il permet d'écrire des modèles concis, lisible et plus conviviaux pour les concepteurs web.

```
{% if is_granted('ROLE_USER') %}
<li class="">
  <a class="nav-item" href="{{ path("commande") }}">Mes commandes</a>
</li>
{% endif %}
```

Debug: Il intègre le composant VarDumper dans les applications Symfony.

Doctrine : Doctrine est l'ORM (Object relationnal Mapping) par défaut du Framework Symfony. Cela permet de synchroniser les entités créées dans l'application avec la base de données. De plus, elle permet de faire le lien entre notre application web et la base de données comme lors de l'insertion d'objet par exemple.

Exemple d'utilisation dans le projet :

```
C:\wamp64\www\hope>php bin/console doctrine:schema:update --force

Updating database schema...

1 query was executed

[OK] Database schema updated successfully!
```

Dans cet exemple, nous avons pu utiliser doctrine afin de mettre à jour la base de données suite à la récupération de projet depuis Git Kraken. Ici, nous voyons que notre projet et la base de données sont synchronisés.

Il a l'avantage d'être compatible avec un maximum de base de données comme MySQL, Oracle ou SQL Server pour ne citer que les plus populaires.

Cependant, cet ORM ne peut effectuer certaines requêtes plus importantes. Il est alors nécessaire de découper la requête en plusieurs parties afin d'obtenir le résultat escompté.

Les requêtes effectuées via Doctrine sont appelées DQL (Doctrine Query Language).

Voici un exemple de DQL possible :

```
$userID = $user->getId();  
$repo = $this->getDoctrine()->getRepository(Commande::class);  
$commande = $repo->findBy(array('utilisateur' => $userID ));
```

Cela équivaut en SQL (Structured Query Language) à

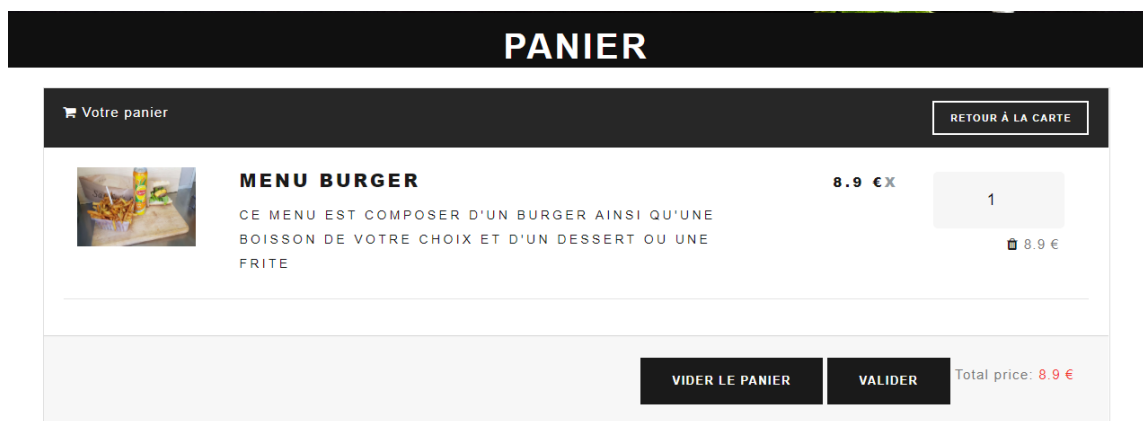
```
SELECT panier FROM `commande` WHERE utilisateur_id = 4
```

8 Présentation de la fonctionnalité la plus significative

8.1 La commande

8.1.1 Le panier

Le panier ayant été réalisé par Geoffrey Olivier, j'en parlerai succinctement ci-dessous afin de me concentrer du passage d'un panier vers une commande enregistrée en base de données.



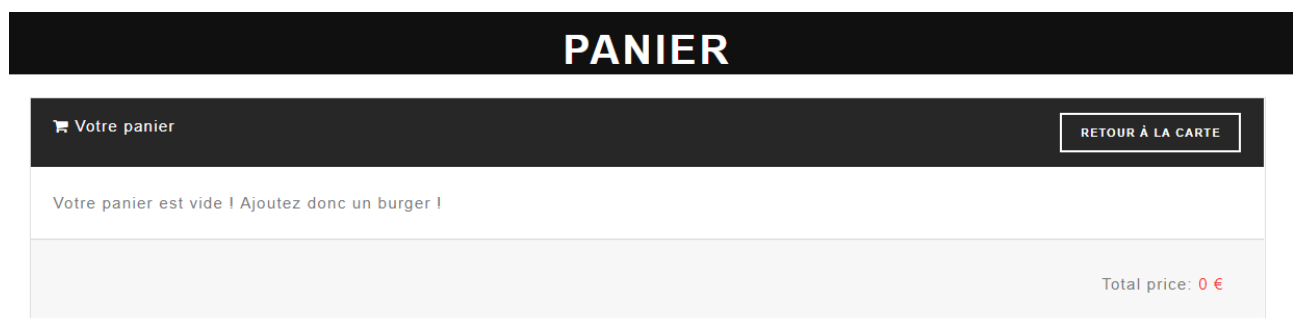
Sur la page Panier, il est possible de visualiser l'intégralité du panier. Vide, il affichera le message suivant « Votre panier est vide ! Ajoutez donc un burger ! ». Sinon il affichera les articles contenus dans le panier.

Il ne sera pas possible de cliquer sur le bouton « commander » grâce à cette condition :

```
{% if panier != null %}

  <a class="btn btn-primary" role="button" href="{{ path('panier.supprime') }}">Vider le
  panier</a>
  <a class="btn btn-primary" role="button" href="{{ path('panier.enregistrement')
  }}">Valider</a>

{% endif %}
```



Cela évite d'avoir de mauvaise manipulation de la part de l'utilisateur et éviter des commandes fantôme. Cela serait une perte de temps pour l'entreprise.
L'utilisateur, connecté ou non, possède une session qui leur est propre. Dans cette dernière, il existe un panier que nous avons créée sous forme de tableau.

```
public function ajouter(SessionInterface $session, Request $request, $id)
{
    if (!$session->has('panier'))
        $session->set('panier', array());
    $panier = $session->get('panier');
    ...
}
```

En effet, nous avons décidé que l'utilisateur connecté ou non puisse avoir un panier. Il peut avoir ainsi un aperçu de son contenu ainsi que du prix total.
Cependant, il devra être connecté pour confirmer son panier et l'enregistrer en « commande ».

8.1.2 Sécurité

```
/**
 * @Route("/profil/commande", name="panier.enregistrement")
 * @param Request $request
 * @return Response
 */

public function enregistrement(EntityManagerInterface $em, Request $request): Response
{
    ...
}
```

Comme la fonction du Controller a pour route /profil/commande, l'utilisateur non connecté sera redirigé vers la page login comme il est précisé dans le fichier « Security.yaml ». Chaque utilisateur connecté possède un rôle user mis à part l'administrateur qui lui, possède un rôle admin.

```
access_control:
- { path: ^/gestionproduit, roles: ROLE_ADMIN }
- { path: ^/profil, roles: ROLE_USER }
- { path: ^/, roles: IS_AUTHENTICATED_ANONYMOUSLY }
- { path: ^/admin, roles: ROLE_ADMIN }
```

8.1.3 Le Controller

```
public function enregistrement(EntityManagerInterface $em, Request $request): Response
{
    $user = $this->getUser();
    $commande = new Commande();
    ...
}
```

Tout d'abord, nous devons récupérer l'utilisateur en cours sur le site et instancier une nouvelle commande pour transférer le contenu du tableau appelé panier dans l'objet commande.

Le principe est de récupérer le panier en session. Chaque visiteur en possède un. Pour cela, nous allons d'abord récupérer la session grâce à l'objet Request puis récupérer le panier que cette dernière possède.

```
$session = $request->getSession();
$panier = $session->get('panier');
...
```

Une fois le panier récupéré, nous hydratons l'instance \$commande avec le contenu du panier dans l'attribut Panier de la table Commande ainsi que l'utilisateur en cours dans l'attribut utilisateur.

```
$commande->setPanier($panier);
$commande->setUtilisateur($user);
```

Par la suite, le panier sera enregistré en base de données dans la commande, la session sera vidée du panier existant et nous en créerons un qui sera vierge.

```
$session->clear();
$session->set('panier', array());
```

Après avoir hydraté cet objet commande, nous la préparons à la sauvegarde en base de données. Puis, nous la sauvegardons grâce à EntityManagerInterface.







```
$em->persist($commande);
$em->flush();
```

8.1.4 Le futur de cette fonctionnalité

Ce projet n'étant pas terminé, il y aura d'autres fonctionnalités que l'utilisateur pourra faire. Avant de valider son panier, il aura la possibilité de choisir l'heure à laquelle il souhaite récupérer sa commande.

Actuellement, l'utilisateur qui aura validé son panier, possèdera une commande. Cette commande est présente en base de données.

+ Options

| | | | | | | | id | utilisateur_id | panier (DC2Type=array) |
|--------------------------|---|--------|---|--------|---|-----------|----|----------------|---------------------------|
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 2 | 4 | a:1:{i:20;i:1;} |
| <input type="checkbox"/> |  | Éditer |  | Copier |  | Supprimer | 3 | 4 | a:2:{i:22;i:1;i:19;i:1;} |

Nous pouvons remarquer que la commande n°2 est attribué à l'utilisateur ayant comme id : 4. Cet utilisateur possède une commande ayant un tableau de produit. Ce tableau prend l'id de l'article et sa quantité.

Les développeurs qui reprendront ce projet pourront ainsi afficher la commande correspondante grâce à une boucle for.

Cette commande affichera l'état de la commande selon des statuts spécifiques défini en amont.

Il pourra supprimer, s'il le souhaite, cette commande si elle n'est pas en préparation par [Oh Ch'ti Breizh](#).

9 Réalisation

Durant ce projet j'ai été amené à travailler sur du Front ainsi que sur du Back-End. Concernant la partie visible de l'utilisateur, cela a été un travail d'équipe pour la grande partie du site. En amont, nous avons discuté, Geoffrey Olivier et moi-même, avec notre maître de stage concernant la disposition des pages et ce qu'elles contiendraient. Par la suite, des modifications ont eu lieu sur toutes les pages du site en cas d'idée nouvelle ou insatisfaction du rendu tout en restant dans l'esprit de la maquette réalisée et des souhaits de Mr Buelens. Concernant la partie Back-End, j'ai eu l'occasion de réaliser plusieurs fonctionnalités que je listerai ci-dessous.

9.1 Front-end

9.1.1 La NavBar

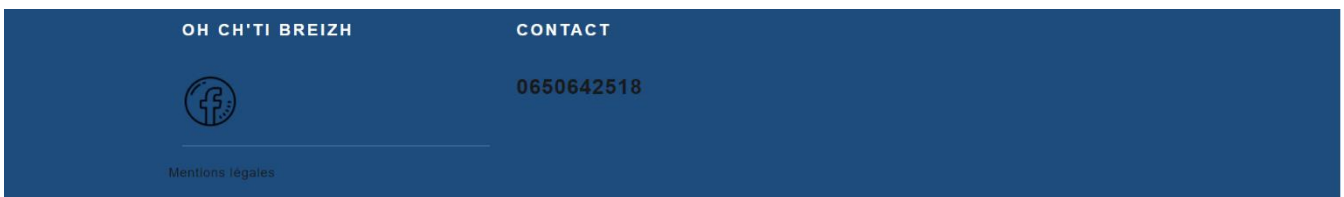
La NavBar a été réalisée en collaboration avec Geoffrey Olivier. Le but étant d'afficher les liens pour accéder aux pages du site, le nom de l'entreprise, ainsi que le panier de l'utilisateur avec le nombre d'article présent dans ce dernier.



Toutes les pages doivent être accessibles depuis la NavBar. Elle a été remaniée par Geoffrey afin d'obtenir le rendu que nous pouvons voir ci-dessus. Elle se situe au-dessus du header.

9.1.2 Le Footer

Le Footer doit permettre d'accéder aux informations telles que la page Facebook de l'entreprise, les mentions légales et le numéro de téléphone de l'entreprise. Ces informations sont essentielles pour l'utilisateur. Elle sera présente sur toutes les pages du site web.



9.1.3 JQuery

J'ai eu l'occasion d'utiliser du jQuery lors du projet. Dans la page d'accueil, l'utilisateur a possibilité de visualiser les lieux de passage du camion dans la semaine en cliquant sur les boutons mises à disposition.

```
<script>
```

```
$(document).ready(function () {
```

```
    $('#mardi').show();  
    $('#mercredi').hide();  
    $('#jeudi').hide();  
    $('#vendredi').hide();  
    $('#samedi').hide();
```

```
...
```

```
    $("#photo1").click(function () {  
        $('#mercredi').hide();  
        $('#jeudi').hide();  
        $('#vendredi').hide();  
        $('#samedi').hide();  
        $('#mardi').show();  
    });
```

```
...
```

```
</script>
```

L'extrait ci-dessus montre le fonctionnement du JQuery. Lorsque l'utilisateur arrive sur la page principal (« main »), la <div> contenant l'id « mardi » s'affichera et les autres <div> ne s'afficheront pas.

Par la suite, lorsque l'utilisateur cliquera sur le bouton contenant l'id « photo1 », alors la <div> contenant « mardi » s'affichera et les autres se fermeront.

La suite du code correspond aux autres boutons cliquables. Lorsque l'utilisateur clique sur un bouton spécifique, une <div> s'ouvre et les autres se ferment. Il y aura autant de <div> que de bouton à disposition.

Mardi Mercredi Jeudi Vendredi **Samedi**



AUJOURD'HUI JE SUIS SITUÉ À ETRELLES !

Adresse: Webhelp, D110, 35370 Étrelles

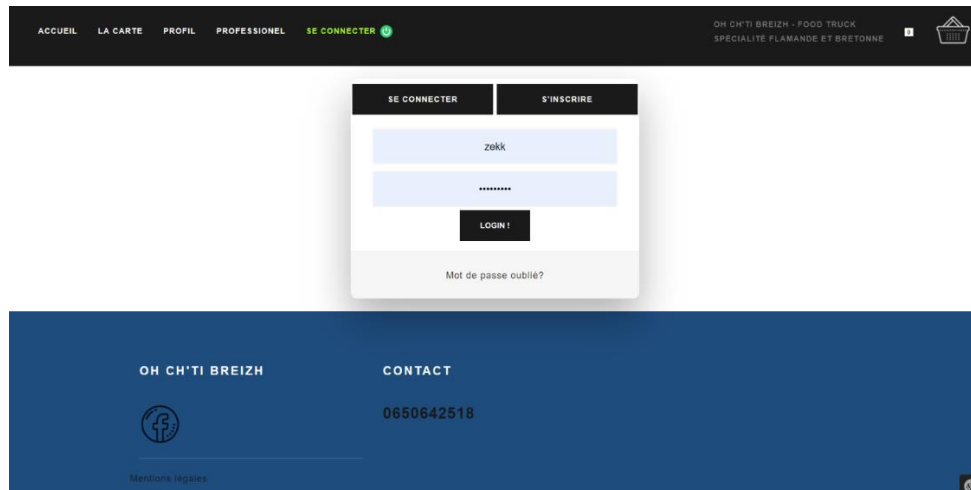
LE MENU

L'utilisateur peut ainsi visualiser ces informations sans recharger la page.

9.2 Front-End & Back-End

9.2.1 Connexion

La page connexion a été la deuxième à être réalisée après la page inscription. Elle permet à l'utilisateur de se connecter afin de profiter de fonctionnalités supplémentaires comme l'envoi de commande et la réservation.



Au niveau Back end, le fichier security.yaml s'occupe de vérifier si les informations entrées par l'utilisateur correspondent avec celle en base de données. Si les informations sont correctes, il pourra accéder à la page principale. Sinon, il sera redirigé sur cette même page tant que les informations ne seront pas bonnes.

De plus, le mot de passe étant encodé en base de données, il est nécessaire d'utiliser ce même procédé pour la connexion.

Ainsi, le mot de passe entré par l'utilisateur devient encodé lui aussi quand il sera comparé avec celui en base de données.

```
security:
  encoders:
    App\Entity\Utilisateur:
      algorithm: bcrypt

  # https://symfony.com/doc/current/security.html#where-do-users-come-from-user-providers

  providers:
    in_memory:
      memory:
        users:
          admin:
            password: password
            roles: 'ROLE_ADMIN'
    in_database:
      entity:
        class: App\Entity\Utilisateur
        property: username

  firewalls:
    dev:
      pattern: ^/(_(profiler|wdt)|css|images|js)/
```

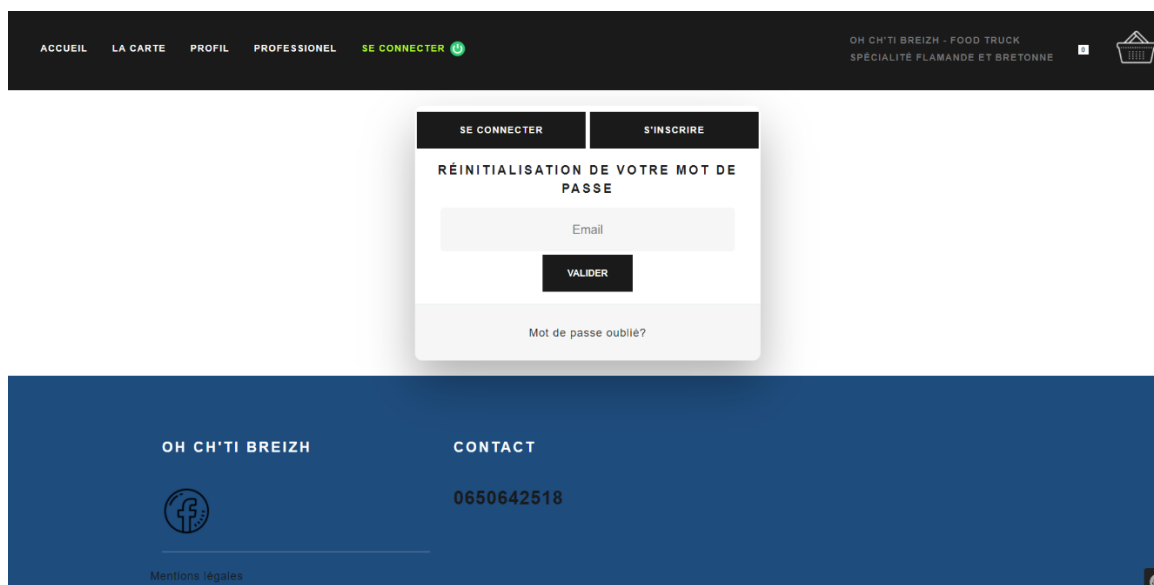
```

security: false
main:
  pattern: ^/
  provider: in_database
  anonymous: ~
  form_login:
    login_path: login
    check_path: login
    default_target_path: main
  logout_on_user_change: true
  logout:
    path: logout
    target: main

```

9.2.2 Mot de passe oublié

La gestion du mot de passe oublié se trouve sur la page connexion. L'utilisateur ne connaissant ou ne se souvenant plus de son mot de passe pourra cliquer sur ce bouton et sera amené à la page suivante :



L'utilisateur entrera l'adresse mail qu'il a fourni lors de son inscription. Si l'adresse mail appartient à un utilisateur, un mail sera envoyé sur cette adresse avec un Token. Un Token est généré automatiquement dans l'entité « utilisateur » en question.

```

$email = $request->request->get('email');

$entityManager = $this->getDoctrine()->getManager();
$user = $entityManager->getRepository(Utilisateur::class)->findOneBy(["email" => $email]);
/* @var $user Utilisateur */

if ($user === null) {
    $this->addFlash('danger', 'Email Inconnu');
    return $this->redirectToRoute('login');
}

```

```

}
$token = $tokenGenerator->generateToken();

try {
    $user->setToken($token);
    $entityManager->flush();
} catch (\Exception $e) {
    $this->addFlash('warning', $e->getMessage());
    return $this->redirectToRoute('main');
}

$url = $this->generateUrl('resetPassword', array("token" => $token),
    UrlGeneratorInterface::ABSOLUTE_URL);
$userEmail = $user->getEmail();
$message = (new \Swift_Message('Mot de passe Oublié'))
    ->setFrom("OhCh'tiBreizh@gmail.com")
    ->setTo($userEmail)
    ->setBody(
        "Bonjour, Vous avez oublié votre mot de passe ? voici le token pour reseter votre
mot de passe : " . $url,
        'text/html'
    );
$mailer->send($message);

$this->addFlash('notice', 'Mail envoyé');

return $this->redirectToRoute('login');
}

```

Un email est donc envoyé grâce au bundle SwiftMailer comprenant une url et le Token. L'utilisateur devra coller le lien sur son navigateur et sera amené à changer son mot de passe.

SE CONNECTER

S'INSCRIRE

RÉINITIALISATION DE VOTRE MOT DE PASSE

Password

ENREGISTRER

Une fois que l'utilisateur saisira son nouveau mot de passe, il sera redirigé vers la page connexion du site.

9.2.3 Profil

La page profil permet à l'utilisateur de vérifier les informations qu'il a entrées lors de l'inscription. Si des modifications sont à faire, il pourra cliquer sur le bouton « modifier ». Le mot de passe ne sera jamais affiché par souci de sécurité.

Prenom:

bobby

Email:

remy_savin@hotmail.fr

MDP:

Votre mot de passe

Telephone:

0664727585

MODIFIER

RETOUR

Le Contrôleur récupère l'utilisateur en cours pour afficher ses informations personnelles. S'il le souhaite, il pourra les modifier en cliquant sur modifier

```
/**
 * @Route("/profil", name="profil")
 * Cette fonction a pour but de récupérer l'utilisateur en cours afin qu'il puisse visualiser ses
 informations sur la page
 */
public function index()
{
    $user = $this->getUser();
    $profilForm = $this->createForm(ProfilType::class, $user);
    return $this->render('profil/profil.html.twig', [
        'controller_name' => 'ProfilController', 'profilForm' => $profilForm->createView()
    ]);
}
```

La route étant /profil, l'utilisateur non connecté sera redirigé sur la page login afin de s'identifier. Une fois ceci fait, il pourra visualiser ses informations.

9.2.4 Modifier profil

L'utilisateur souhaitant modifier son profil pourra le faire sans aucun mal sur cette page. Cependant, le pseudo étant nécessaire à la connexion et unique, il ne pourra pas le modifier. Seul les informations présentes sur cette page sont modifiables. Les champs sont déjà préremplis par les informations enregistrées en base de données. Ainsi, si l'utilisateur a un seul changement à effectuer, il n'aura pas à remplir le reste. Il devra par la suite entrer son mot de passe afin de valider le changement.

Dans le Contrôleur, nous allons récupérer l'utilisateur en cours, préparer le formulaire et nous préparer à récupérer les informations contenues dans ce dernier.

```
/**
 * @Route("/profil/modifierProfil", name="modifierProfil")
 * Cette fonction a pour but de modifier les données de l'utilisateur. Il ne pourra accéder à ce
 controller que si
```

```

* il est connecté, sinon il sera renvoyé vers la page login
*/
public function modifierProfil(EntityManagerInterface $em, Request $request,
UserPasswordEncoderInterface $encoder)
{
    //récupération de l'utilisateur en cours
    $user = $this->getUser();

    $modifierProfilForm = $this->createForm(ModifierProfilType::class, $user);
    $modifierProfilForm->handleRequest($request);
    ...

```

Si ce formulaire est soumis et qu'il est valide, nous allons vérifier que le mot de passe entré par l'utilisateur est bien celui que nous avons en base de données.

Si le mot de passe ne correspond pas, il retournera sur cette page avec un message lui indiquant l'erreur que nous rencontrons.

Si le mot de passe correspond, le Controller se chargera de récupérer les informations et l'enverra en base de données, écrasant les données déjà existantes.

```

...
if ($hash !== true) {
    $this->addFlash('alert', 'Le Mot de Passe ne correspond pas avec celui que nous
avons en BDD !');
    return $this->render('profil/modifierProfil.html.twig', [
        'controller_name' => 'ModifierProfilController', "modifierProfilForm" =>
$modifierProfilForm->createView()]);
    } else {

        $em->persist($user);
        $em->flush();
        $this->addFlash('success', 'La modification a bien été enregistrée !');
        return $this->redirectToRoute('profil');
    }
}

return $this->render('profil/modifierProfil.html.twig', [
    'controller_name' => 'ModifierProfilController', "modifierProfilForm" =>
$modifierProfilForm->createView()

]);
}
...

```

9.2.5 Contact

La page contact permettra à l'utilisateur d'envoyer des messages à l'entreprise.

Pour l'instant, l'adresse mail de l'entreprise n'existant pas encore, j'en ai créé une spécialement pour cette utilisation. Par la suite, il ne restera plus qu'à changer l'adresse mail par celle d' « Oh Ch'ti Breizh »

L'utilisateur entrera les informations essentielles à l'envoi du mail et choisira le motif de son message.

CONTACTEZ LE FOOD TRUCK

Nom

Email

Prenom


Telephone

Objet :

mariage

Message

☐ Je ne suis pas un robot



reCAPTCHA

Confidentialité - Conditions

VALIDER

RETOUR

Nous allons donc devoir effectuer quelques préparations pour cela. Intéressons-nous au Controller :

```

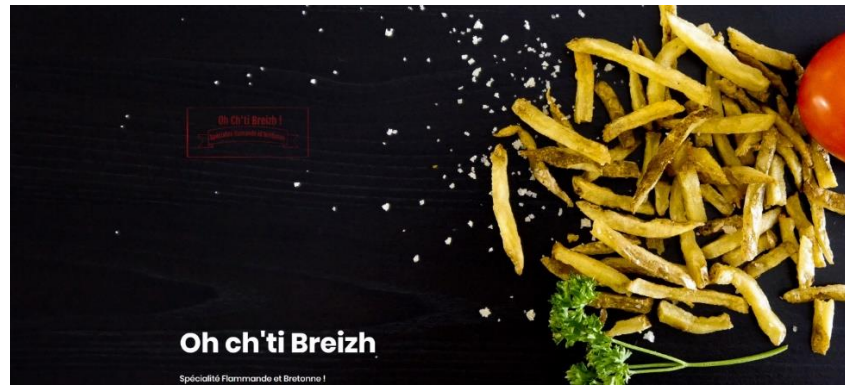
/**
 * @Route("/contact", name="contact")
 * Cette fonction a pour but de récupérer les éléments inscrits par l'utilisateur et d'envoyer un mail à
 * Oh Ch'ti Breizh
 * une fois que le formulaire est soumis, qu'il est valide et que le recaptcha est coché.
 * Un recaptcha est présent et suit la même logique que dans le controller connexion/register
 *
 */
public function contact(Request $request, \Swift_Mailer $mailer)
{
    $recaptcha = new ReCaptcha('6LetZbEUA AAAAIC0A8lfxY3WDCvuWySaeJHm_6aT');
    $contact = new Contact();
    $contactForm = $this->createForm(ContactType::class, $contact);
    $contactForm -> handleRequest($request);
    $resp = $recaptcha->verify($request->request->get('g-recaptcha-response'), $request->getClientIp());
}

```

9.3 Autre

9.3.1 Photo

Nous avons réalisé également les photos des produits de l'entreprise. Le but étant d'attirer l'œil et donner envie au client à découvrir les produits de l'entreprise.



La photo ci-dessus a nécessité du travail pour obtenir ce type de rendu. Après avoir pris environ une centaine de photos, nous avons filtré celles qui avaient des défauts (flou par exemple) afin de n'en garder qu'une dizaine. Puis après modification via un logiciel de retouche, nous avons pu obtenir le résultat escompté.

Cette photo est présente sur plusieurs pages de notre site juste en dessous de la NavBar.

Pour les autres photos, nous voulions les réaliser directement au camion. Ainsi, nous avons les produits à portée de main.

10 Présentation du jeu d'essai

L'application web « Oh Ch'ti Breizh » permet à l'utilisateur de connaître d'avantage le Food-truck ainsi que les services proposés.

L'utilisateur peut ainsi connaître les lieux où passe le Food truck dans la semaine.

De plus, il peut voir la carte, contacter Oh Ch'ti Breizh pour avoir des informations supplémentaires, voir même de demander à l'entreprise de s'occuper de la restauration lors d'un évènement.

Enfin, il pourra également, après avoir vu la carte, commander en ligne. Il pourra ainsi récupérer sa commande et payer au camion.

Cependant, cette dernière fonctionnalité nécessite pour l'utilisateur d'être enregistré sur le site web.

Nous allons nous concentrer sur la page s'inscrire qui nécessite plusieurs sécurités précises.

Cette page permet à l'utilisateur de s'inscrire sur le site afin de profiter des services en ligne du Food-Truck. Une fois inscrit, il sera ramené directement sur la page login afin de saisir son identifiant et son mot de passe.

10.1 Inscription

The screenshot shows a web browser window with the URL `localhost:8082/ohpe/public/register`. The page has a dark header with navigation links: ACCUEIL, LA CARTE, PROFIL, MES COMMANDES, PROFESSIONNEL, SE DÉCONNECTER (with a red power icon), and OH CH'TI BREIZH - FOOD TRUCK SPECIALITÉ FLAMANDE ET BRETONNE. A shopping cart icon is on the right. The main content area has two tabs: SE CONNECTER and S'INSCRIRE (selected). The registration form includes fields for pseudo (filled with 'zekk'), Email, Mot de passe (masked with dots), and Telephone. Below the fields is a checkbox for 'J'accepte les conditions générales de ventes et d'utilisation'. At the bottom is a reCAPTCHA widget with the text 'Je ne suis pas un robot' and a 'VALIDER' button.

Concernant les informations demandées, M. Buelens aura besoin des informations comme le numéro de téléphone ou le mail pour contacter l'utilisateur en cas de besoin.

De plus, le pseudo et le mot de passe permettra à l'utilisateur de se connecter au site.

Il n'est donc pas nécessaire de demander d'autres informations, nous souhaitons avoir les informations juste nécessaires au bon fonctionnement du site.

Son pseudo sera unique. Il ne pourra pas le changer par la suite.

Le mot de passe lui sera crypté avant d'être envoyé en base de données.

Il sera nécessaire de cocher la case concernant les conditions générales de vente. L'utilisateur aura la possibilité d'y accéder directement en cliquant dessus ou y accéder dans le Footer du site en cliquant sur l'onglet « Mentions légale ».

Cette dernière est très importante car elle doit protéger l'entreprise. Mr Buelens se chargera de la création des conditions générales de vente.

De plus, nous pouvons apercevoir un ReCAPTCHA juste avant le bouton valider. Pour que l'inscription fonctionne, l'utilisateur devra cliquer dessus.

10.1.1 Création du Controller

Nous allons donc créer un Controller qui s'occupera de la fonction inscription. Pour cela, nous allons devoir utiliser Cmdr, nous rendre dans le dossier du projet et effectuer la commande suivante :

```
C:\wamp64\www\hope>php bin/console make:controller
```

Une fois le Controller créé, nous allons devoir lui donner un nom de route et définir les actions automatiques qu'il fera lorsque l'utilisateur utilisera cette fonction.

```
/**
 * @Route("/register", name="register")
 * Cette fonction a pour but d'enregistrer un nouvel utilisateur en base de données
 * et d'encoder son mot de passe automatiquement
 */
public function register(EntityManagerInterface $em, Request $request,
UserPasswordEncoderInterface $encoder)
{
    // on instancie un nouvel utilisateur et on lui donne le rôle (Rôle User). Il pourra accéder aux
    pages définies dans
    // security.yaml
    $user = new Utilisateur();
    $user->addRole("ROLE_USER");
    ...
}
```

Cette fonction register aura besoin en paramètre :

EntityManagerInterface : C'est un composant de L'ORM Doctrine. Il permet de traduire de manière simple les requêtes SQL.

Request : c'est un objet qui va permettre de construire une réponse en fonction de la requête.

UserPasswordEncoderInterface : C'est un service intégré à Symfony qui permet d'encoder les mots de passe efficacement. Il sera nécessaire d'implémenter le UserInterface dans l'entité Utilisateur pour l'utiliser.

```
class Utilisateur implements UserInterface
```

10.1.2 Création du formulaire

Nous allons créer un formulaire comprenant les informations que devra remplir l'utilisateur. Pour cela, nous allons utiliser notre console d'exécution Cmdr afin de le créer. Nous l'appellerons InscriptionType.

```
C:\wamp64\www\hope>php bin/console make:form
```

Une fois créée, nous allons paramétrer notre formulaire et gérer les attributs de chaque champ.

```

public function buildForm(FormBuilderInterface $builder, array $options)
{
    $builder
        ->add('telephone', TelType::class, ['label'=>'Telephone:', 'attr'=>['required' => true]])
        ->add('password', PasswordType::class, ['mapped' => false, 'label'=>'Mot de passe:',
'attr'=>['required' => true]])
        ->add('username', TextType::class, ['label'=>'pseudo:', 'attr'=>['required' => true]])
        ->add('email', EmailType::class, ['label'=>'Email:', 'attr'=>['required' => true]])

    ;
}

public function configureOptions(OptionsResolver $resolver)
{
    $resolver->setDefaults([
        'data_class' => Utilisateur::class,
    ]);
}

```

Ainsi l'inscription devra nécessiter le remplissage de ces champs pour que l'inscription fonctionne et soit complète.

10.1.3 Préparation du Controller

Afin de créer un nouvel utilisateur, nous allons en instancier un, grâce à l'instanciation.

```
$user = new Utilisateur();
```

Nous allons hydrater cet Objet et lui donner un rôle. La personne qui consulte le site internet sans se connecter est considérée comme anonyme et sera limitée d'accès. Ici, nous lui donnons le rôle « user ».

```
$user->addRole("ROLE_USER");
```

Cet utilisateur une fois connecté pourra ainsi naviguer sur les pages autorisées par le fichier « security.yaml ».

access_control:

```

...
- { path: ^/profil, roles: ROLE_USER }
- { path: ^/, roles: IS_AUTHENTICATED_ANONYMOUSLY}

```

Nous remarquons, ci-dessus, que l'utilisateur non connecté a pour rôle « IS_AUTHENTICATED_ANONYMOUSLY ». Il ne pourra donc pas consulter les pages commençant par http://www.../profil .

```
$inscireForm = $this->createForm(InscriptionType::class, $user);
```

```

//Récupération automatique des données du formulaire dans l'entité Utilisateur
$inscireForm->handleRequest($request);

```

Ici, nous allons préparer le formulaire à l'envoi et nous préparer à récupérer les informations entrées par l'utilisateur dans ce formulaire.

10.1.4 Le ReCAPTCHA

Maintenant, nous allons nous occuper du reCAPTCHA. Pour la mettre en place, nous allons tout d'abord l'installer via notre console d'exécution.

```
C:\wamp64\www\hope>composer req google/recaptcha
```

Ensuite via le Controller, nous allons instancier le reCAPTCHA et lui donner la clé secrète. Cette clé ne sera pas visible par l'utilisateur comme elle fait partie du Controller. Puis nous allons vérifier que l'utilisateur a cliquer sur notre reCAPTCHA.

```
// on instancie le recaptcha en lui fournissant la clé secrète. La clé publique sera dans le twig
$recaptcha = new ReCaptcha('6LetZbEUAAAAAIC0A8IfxY3WDCvuWySaeJHm_6aT');

//Vérification si la case est cochée du recaptcha
$response = $recaptcha->verify($request->request->get('g-recaptcha-response'), $request->getClientIp());
```

Nous allons vérifier que l'utilisateur à soumis le formulaire, qu'il est valide et que le reCAPTCHA a bien été validé.

Si c'est le cas, nous allons pouvoir récupérer les éléments dont nous avons besoin.

10.1.5 Encodage et envoi en base de données

Sachant que les utilisateurs ont, pour la majorité, le même mot de passe pour plusieurs sites, Cela serait désastreux pour l'utilisateur et préjudiciable pour l'entreprise car en cas de vol de donnée, l'utilisateur pourrait se retourner contre l'entreprise. Il est tout à fait possible d'encoder le password de l'utilisateur manuellement grâce à :

```
C:\wamp64\www\hope>php bin/console security:encode password

Symfony Password Encoder Utility
=====

-----
Key                Value
-----
Encoder used       Symfony\Component\Security\Core\Encoder\BCryptPasswordEncoder
Encoded password    $2y$13$nLT6FGho9Hxv0G1JLE46mes19fsEFH3ywfOVDJ.ITXx0r93NV1ZQ.
-----

! [NOTE] Self-salting encoder used: the encoder generated its own built-in salt.

[OK] Password encoding succeeded
```

Ici password est encodé grâce à la méthode BCrypt. Nous pouvons créer un utilisateur en base de données et insérer le mot de passe encodé. Cette action est possible mais nécessite à l'administrateur de connaître le mot de passe de l'utilisateur. De plus, cette action est fastidieuse. Nous avons donc besoin de l'automatiser.

Nous allons récupérer ce qu'il a entré dans l'input ayant pour nom « password ». Puis nous allons l'encoder et l'enregistrer dans une variable.

```
//On vérifie que le formulaire est soumis, qu'il est bien valide et que le recaptcha est ok.
if ($inscrireForm->isSubmitted() && $inscrireForm->isValid() && $response->isSuccess()) {

    // on récupère le mot de passe inscrit par l'utilisateur et on l'encode
```

```
$passwordUnencode = $inscrireForm->get('password')->getData();  
$hash = $encoder->encodePassword($user, $passwordUnencode);
```

Ainsi nous allons pouvoir hydrater à nouveau notre objet \$user et lui envoyer le mot de passe crypté.

Il nous reste plus qu'à préparer l'envoi en base de données par l'utilisateur et envoyer les informations transmises par l'utilisateur.

```
// on fournit à l'objet utilisateur le mot de passe encodé  
$user->setPassword($hash);  
$em->persist($user);  
$em->flush();
```

11 Sécurité

11.1 Le reCAPTCHA

11.1.1 Pourquoi installer ReCAPTCHA

Le ReCAPTCHA est un API mis au point par Google. Une API (Application Programming Interface) est une interface de programmation qui permet de se brancher sur une application pour échanger des données. Ici, Google autorise les applications à utiliser son programme.

Son but est de protéger notre application web contre les attaques de type « brute-force » ou de spams par des hacker ou des bots.

Un hacker est un spécialiste informatique qui recherche les moyens de contourner les protections afin de détourner, par exemple, le but principal d'un site internet. Il peut agir par curiosité, en recherche de notoriété ou contre rémunération.

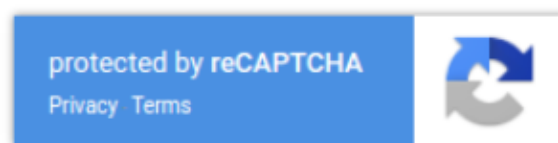
Un bot est un logiciel informatique. Il peut être utilisé dans plusieurs domaines comme les jeux vidéo par exemple. Il suit un script précis. Dans notre cas, nous souhaitons sécuriser au maximum notre application web car il est nécessaire de savoir que 50% de tout le trafic est généré par des bots.

Dans notre situation, les bots sont beaucoup plus problématiques. En effet, ils ont la particularité de pouvoir effectuer un grand nombre de requête sur un laps de temps très court.

11.1.2 Choix du type de reCAPTCHA

Il est possible de choisir entre trois types de captcha dans la V2 :

- La première où l'utilisateur doit reconnaître des panneaux, feu de signalisation... Fastidieux pour l'utilisateur car même un humain aura peut-être besoin de plusieurs tentatives avant de la valider.
- La deuxième, celle présente dans l'image ci-dessus et présente sur le site, nécessite que l'utilisateur clique dessus. Simple et rapide, il y a peu de chance d'échec.
- La troisième, est invisible. Cet API se lancera dès qu'un utilisateur clique sur un bouton existant sur le site. Il laissera ainsi un logo présent pendant toute la navigation du site.



A l'heure actuelle, la V3 est possible d'être utilisée. Cependant ayant eu quelques difficultés à mettre en place la V2, j'ai dû me résoudre à mettre de côté l'installation de la V3 par manque de temps.

11.1.3 Installation

Mon choix s'est porté sur la V2 avec bouton à cocher . Il est facile à mettre en place et j'ai uniquement 2 pages à sécuriser : la page Inscription et la page Contact.

Pour pouvoir le mettre en place, il est nécessaire de se rendre sur le site <https://www.google.com/recaptcha/> . Il permet d'enregistrer le nom de domaine de notre application web et ainsi nous fournir une clé publique et une clé secrète.

La clé publique se trouvera sur la page Twig.

```
<div class="g-recaptcha recaptcha" data-sitekey="6LetZbEUAAAAAJoul08x5JV_6EQTiWLowVAAAw" ></div>
```

La clé secrète sera dans le Controller.

```
$recaptcha = new ReCaptcha('6LetZbEUAAAAAIC0A8IfxY3WDCvuWySaeJHm_6aT');
```

A noter que pour un site en localhost, cela nécessite l'installation d'une extension sur son navigateur. Pour ma part j'ai utilisé l'extension « Virtual Host » afin de pouvoir faire fonctionner le ReCAPTCHA en localhost. Le faire fonctionner ainsi, a été laborieux car je pensais que c'était l'installation qui n'était pas bonne. Il m'a fallu quelques recherches sur des forums spécialisés pour trouver l'information car elle n'était pas présente dans la documentation de Symfony, ni sur le site <https://www.google.com/recaptcha/>.

11.2 Le hashage

11.2.1 Pourquoi hasher le mot de passe

Une règle primordiale : nous ne devons pas conserver de mot de passe en « clair » en base de données. Si cette dernière était piratée, le hacker ou le bot pourrait récupérer toutes les informations s'y trouvant.

Ainsi, lors de l'inscription de l'utilisateur, il est possible d'automatiser le processus de hachage

Le hashage de mot de passe est l'une des pratiques de sécurité les plus basiques qui doit être effectuée.

La non-mise en application de cette pratique entrainerai la perte de chaque mot de passe si la base de données est compromise.

Ainsi, il pourrait être utilisé sur cette même application web et sur d'autres site web si l'utilisateur utilise le même. Malheureusement, la plupart des utilisateurs utilisent encore aujourd'hui un seul et même mot de passe.

11.2.2 Comment sécuriser le mot de passe

Auparavant le hashage pouvait se faire via le hash md5 ou sha1. Mais les standards de sécurité évoluent avec le temps.

Dans ce projet, nous avons utilisé l'encodage BCrypt en le déclarant dans le fichier security.yaml.

Grâce à Symfony, UserPasswordEncoderInterface, nous pouvons utiliser cette méthode afin de crypter le mot de passe.

```
$passwordUnencode = $inscrireForm->get('password')->getData();  
$hash = $encoder->encodePassword($user, $passwordUnencode);
```

C'est une méthode très simple mais qui a le mérite d'être très efficace.

L'utilisateur entrera son mot de passe en clair et avant de l'envoyer en base de données, il suffit d'effectuer cette manipulation, de récupérer cette valeur et de la hasher.

11.3 Les expressions régulières

11.3.1 Définition

Une expression régulière est une chaîne de caractère qui décrit, selon une syntaxe précise, un ensemble de chaîne de caractères possible. Elles sont également appelées regex (de l'anglais regular expression).

C'est un système très ingénieux permettant de nombreuses utilisations :

- Il permet d'analyser différentes chaînes de caractères et ainsi vérifier que la chaîne correspond bien à ce que l'on demande

- Il permet également de retrouver un mot, voir même une phrase complète dans un texte, en fonction d'un modèle que l'on aura défini : Le pattern

11.3.2 Choix d'utilisation

Pour ce projet, nous avons utilisé les Regex afin de vérifier que les informations entrées par l'utilisateur correspondaient bien avec ce que l'on souhaitait obtenir.

```
/**
 * @ORM\Column(type="string", length=255)
 * @Assert\Regex(
 *     pattern = "/^[0][2,6,7,9][0-9]{8}+$/i",
 *     htmlPattern = "[0][2,6,7,9][0-9]{8}+ $"
 * )
 */
private $telephone;
```

Dans l'exemple ci-dessous, nous avons le modèle (ou pattern) qui a été défini :

^ : ancré en début de chaîne

[0] : le premier numéro attendu est un 0

[2, 6, 7, 9] : le deuxième numéro ne peut être que 2, 6, 7, 9. Ainsi l'utilisateur peut laisser son numéro de téléphone fixe (02 ou 09) ou mobile (06 ou 07). Cela évitera à certains utilisateurs mal intentionnés de mettre un numéro de téléphone surtaxé commençant par 08 par exemple.

[0-9]{8} : les 8 derniers chiffres seront entre 0 et 9.

Grâce à ce pattern, nous avons un numéro à 10 chiffres spécifique que l'utilisateur pourra entrer.

Il était possible de le mettre en place pour gérer le password. Cependant, après recherche, il était déconseillé de le mettre en place par souci de maintenance.

Voici un exemple de regex pour mot de passe :

```
^S*(?=\S{8,})(?=\S*[a-z])(?=\S*[A-Z])(?=\S*[\d])\S*$
```

Cela oblige les utilisateurs à entrer un mot de passe de minimum de 8 caractères contenant au moins une lettre majuscule et au moins une lettre en minuscule ainsi qu'un numéro.

Ce type de mot de passe est difficile à retenir pour un humain tandis qu'un ordinateur aura plus de facilité. Il serait ainsi plus judicieux de proposer une jauge de force plutôt que de restreindre les utilisateurs.

De plus, nous pouvons conseiller les utilisateurs à vérifier leurs mots de passe et leurs password sur ce site <https://haveibeenpwned.com> afin qu'ils vérifient par eux-mêmes s'ils sont compromis ou non.

12 Veille effectuée

12.1 Recherche

12.1.1 Contexte

Lors du projet, j'ai eu la nécessité de recourir très souvent à la documentation présente sur le site www.symfony.com,
Cependant, la documentation fournie n'est quelque fois pas assez explicite.
J'ai eu de ce fait recours à d'autres sites anglophones afin d'obtenir d'avantage d'informations.

12.1.2 Phase de recherche

La mise en place du reCAPTCHA, malgré sa simplicité de mise en place d'après le site officiel <https://developers.google.com/recaptcha/docs/display>, a nécessité quelques recherches supplémentaires.

De ce fait, sur le moteur de recherche j'ai d'abord tapé les mots clefs suivant :

« ReCAPTCHA Symfony »

J'ai regardé les vidéos proposées comme « créer un bundle reCAPTCHA » mais la documentation initiale expliquait que la mise en place était simple. Bien qu'intéressant mais compliqué, j'ai continué ma recherche sur les autres sites.

Les deux premiers sites étaient intéressants mais ils manquaient des éléments d'explications.

Le troisième site contenait tout ce dont j'avais besoin. L'installation était expliquée, la mise en place point par point, tout y était.

Le site qui m'a été très utile est :

<https://ourcodeworld.com/articles/read/136/implementing-google-recaptcha-on-a-form-in-symfony-3>

Par la suite, vous trouverez le contenu original du site internet puis la version traduite.

12.2 Traduction

12.2.1 Version originale

What is reCAPTCHA

ReCAPTCHA is a free service that protects your website from spam and abuse. ReCAPTCHA uses an advanced risk analysis engine and adaptive CAPTCHAs to keep automated software from engaging in abusive activities on your site. It does this while letting your valid users pass through with ease.

ReCAPTCHA doesn't depend solely on text distortions to separate man from machines. Rather it uses advanced risk analysis techniques, considering the user's entire engagement with the CAPTCHA, and evaluates a broad range of cues that distinguish humans from bots. reCAPTCHA knows when to be easy on people and hard on bots. Easy CAPTCHAs are a breeze for legitimate users.

Requirements

To use reCAPTCHA on your website, you'll need to :

- Register your website in the reCAPTCHA homepage here

After your registration of your website, you'll be redirected to the admin panel, there will be all the websites that you have. Click on the one you want to use.

You'll find a menu with the keys, client-side integration example and server-side integration example. Be sure to save your site key and secret key as we'll need them later.

That should be enough to handle the client-side integration.

- Include the google recaptcha php library to your project using composer.

To include the google recaptcha php library, require it directly in the composer console executing :

```
composer require google/recaptcha "~1.1"
```

That should be enough to handle the server-side integration, let's get started !

Implementing

For this example, we are going to use a plain contact html form, you can obviously use a self-made symfony form or a crud-generated form. Just be sure to add the new field inside the form.

```
<div class="g-recaptcha" data-sitekey="mypublicdata-sitekey-ofthe-google-panel"></div>
```

No matter the way you render your form, just be sure that the widget is inside the form. Now the widget should appear in your form already without any problem if your data-sitekey value is correct and matches with your website. With the captcha api, this div will be a form element inside the form, this will contain a value that we will retrieve with php on the submit event to validate it.

Now you only need to process the submit action in the controller, where your form is pointing to. »

12.2.2 Version traduite

Qu'est-ce que ReCAPTCHA

ReCAPTCHA est un service gratuit qui protège votre site web contre le spam et les abus. ReCAPTCHA utilise un moteur d'analyse de risque avancé et des CAPTCHA adaptatifs pour empêcher les logiciels automatisés de se livrer à des activités abusives sur votre site. Il le fait tout en laissant vos utilisateurs valides passer avec facilité.

ReCAPTCHA ne dépend pas uniquement de distorsions de texte pour séparer l'homme de la machine. Au lieu de cela, il utilise des techniques avancées d'analyse des risques, prenant en compte l'engagement total de l'utilisateur avec la CAPTCHA, et évalue un large éventail d'indices qui distinguent les humains des robots. Repatcha sait quand être simple avec les gens et difficile avec les bots. Les captchas faciles sont un jeu d'enfant pour les utilisateurs légitimes.

Exigences

Pour utiliser reCAPTCHA dans votre projet web, vous aurez besoin :

- D'enregistrer votre site web sur la page d'accueil reCAPTCHA ici.

Après l'enregistrement de votre site, vous allez être redirigé au panneau d'administration, qui listera l'ensemble des sites que vous possédez. Cliquez sur celui que vous souhaitez utiliser.

Vous trouverez un menu avec les Clés coté client et coté serveur. Veillez à bien sauvegarder votre clé publique et votre clé secrète, nous allons en avoir besoin plus tard.

Cela est suffisant pour gérer l'intégration coté client.

- Inclure la librairie PHP Google reCAPTCHA à votre projet en utilisant composer .

Pour inclure la librairie Google reCAPTCHA , entrez la ligne suivante dans votre console.

```
composer require google/recaptcha "~1.1"
```

Cela devrait être suffisant pour gérer l'intégration coté serveur, commençons !

L'implémentation

Pour cet exemple nous allons utiliser un formulaire de contact en html pur. Vous pouvez évidemment utiliser un formulaire fait sous Symfony, ou un formulaire auto-généré par le CRUD. Veillez simplement à bien intégrer la ligne suivante au sein de votre formulaire :

```
<div class="g-recaptcha" data-sitekey="mypublicdata-sitekey-ofthe-google-panel"></div>
```

Peu importe la façon dont vous générez votre formulaire, soyez juste certain que le Widget soit bien intégré au formulaire.

A présent le Widget devrait déjà apparaître au sein du formulaire, (si la clef renseignée est correcte et correspond à celle de votre site web). Avec l'api Captcha, cette div sera un élément formulaire au sein de votre formulaire, il va contenir une valeur que nous allons récupérer avec PHP pour le valider au moment de la soumission de celui-ci. A présent vous avez seulement besoin de procéder à l'action de traitement au sein du Controller vers lequel pointe le formulaire.

13 Conclusion

Ce projet a été réalisé avec beaucoup d'enthousiasme et beaucoup de sérieux. Cependant, je ne m'attendais pas à certaines difficultés techniques telles que la mise en relation avec la base de données qui ne s'effectuait pas ou git kraken qui était « opérationnel » mais qui n'enregistrait pas notre travail.

Ce projet a été très formateur pour moi. Il m'a obligé à rechercher les informations et à comprendre mes propres erreurs.

En milieu de stage, malgré notre travail, nous avons réalisé que le projet ne pourrait être mise en ligne comme nous le souhaitions. En effet, cet e-commerce nécessitait des conditions générales de vente de qualité pour l'entreprise « Oh Ch'ti Breizh » et nécessitait à M. Buelens de voir un avocat pour l'élaboration de cette dernière.

De plus, la conception du projet a pris énormément de temps et nous a obligé à revoir la quantité de fonctionnalités mise en place par nos soins à la baisse.

Il a nécessité de fournir une documentation pour les futurs développeurs qui reprendront le projet. Chaque Controller possède des commentaires associés expliquant notre travail.

Un document sera fourni pour les développeurs expliquant les fonctionnalités mises en place et celles restantes comme l'affichage de la commande ou la réception des commandes par l'administrateur. Elle sera présente en annexe (voir annexe p.43). De plus, l'intégralité du travail sera sur GitHub avec le cahier des charges, la partie conception et le script SQL. Au cas où, il sera également transmis à M. Buelens sur clé USB.

J'ai été très content de travailler sur ce projet durant ce stage. Mais une part de déception subsiste de ne pas l'avoir entièrement fini au vu de mon investissement personnel.

Cependant, je ne reste pas sur ce point négatif. Il m'a permis de mettre en application mes connaissances théoriques.

J'espère que les développeurs à venir pourront comprendre aisément notre travail et peut être que, une gestion de paiement sera envisagée par M. Buelens.

14 Annexes

Pour la poursuite du projet Oh Ch'ti Breizh

Le but de cette application web est de permettre à l'utilisateur de commander des produits du Food-Truck afin de les récupérer, après paiement, au camion.

Actuellement le site web est toujours en projet et des fonctionnalités supplémentaires sont à ajouter. Je les listerais ci-dessous.

Le cahier des charges se trouve dans ce rapport de stage page 8.

Nous avons été 2 sur ce projet pendant une durée de 2 mois du 24 Juin 2019 au 26 août 2019.

Les Controllers sont commentés afin de comprendre le but recherché. Si besoin, des commentaires supplémentaires ont été rajoutés pour expliquer notre façon de penser et de faire.

Le design ayant été validé par Mr Buelens, gérant de l'entreprise Oh Ch'ti Breizh, il convient que le design présent sur le site devra être respecté au maximum. Des changements sont cependant tout à fait possibles.

Les photos présentes sur le site ont été réalisés par nos soins et l'entreprise Oh Ch'ti Breizh en est détentrice.

Les fonctionnalités mis en place :

- Inscription
L'utilisateur peut s'inscrire sur cette page. Elle est sécurisée par un reCAPTCHA V2. Des Regexs ont été mis en place dans l'entité utilisateur afin de renforcer la sécurité.
- Connexion, géré par security.yaml
L'utilisateur devra, pour commander ou accéder à sa page profil, se connecter via cette page.
- Profil Utilisateur
L'utilisateur pourra visualiser ses informations si elles sont correctes. Il peut soit choisir de les modifier en cliquant sur le bouton modifier ou retourner sur la page principale. Le mot de passe n'est nullement visible par sécurité.
- Modifier Profil
L'utilisateur aura possibilité de modifier son adresse mail, numéro de téléphone. Le pseudo ne pourra pas être changé car il est nécessaire à la connexion et est unique.
- Contact
L'utilisateur, qu'il soit connecté ou non, pourra envoyer un message à Oh Ch'ti Breizh. Elle est sécurisée par un reCAPTCHA V2 afin d'éviter les spams. Cette fonctionnalité fonctionne grâce au bundle SwiftMailer.
- Panier
Nous avons souhaité que l'utilisateur, même non connecté, puisse avoir un panier. Ce panier est stocké sous forme d'array dans la session de l'utilisateur. La validation nécessite que le panier comprenne au moins un article et que l'utilisateur est connecté
- Commande
L'utilisateur connecté pourra valider son panier ce qui l'enverra en base de données dans « commande ».

Fonctionnalités restantes :

- Commande : afficher la commande du client. L'administrateur doit pour réceptionner les commandes du jour et mettre un statut « non défini » par default, « en cours », « prêt » ou

« refus »

Ce statut doit être automatique et l'utilisateur devra pouvoir visualiser le changement de statut sans rechargement de la page

- Panier : L'utilisateur pourra choisir un créneau horaire défini pour récupérer sa commande. Il ne peut y avoir que 8 personnes toutes les 10 minutes par créneau. De plus, il pourra choisir la date de récupération de la commande. Elle devra être visible dans la commande
- Détail Article : Les articles proposés par Oh Ch'ti Breizh seront présents dans la page « la carte ». Il sera possible de personnaliser les sandwiches et d'ajouter des suppléments contre surcoût. Il pourra également retirer des ingrédients. Cela nécessite d'ajouter une entité ingrédient en base de données.
- Gestion des Stocks
Le stock sera séparé en 2 : un stock pour les commandes web et un stock pour les commandes au camion.
- Les lieux de passage
L'admin pourra modifier les lieux de passage si besoin. Cela modifiera la page streetview.
- Insérer un streetview sur la page d'accueil qui indiquera à l'utilisateur les lieux de passage du camion tout au long de la semaine.
- Ajout de message
L'admin pourra ajouter un message qui sera visible par les utilisateurs allant sur le site.
- Demander la certification HTTPS

La gestion de paiement n'est, à l'heure actuelle, pas à l'ordre du jour. Oh Ch'ti Breizh souhaite que le paiement se fasse directement au camion.

D'autres fonctionnalités pourront être mis en place. Parlez-en directement à Oh Ch'ti Breizh pour validation.